giganci programowania

Materiały powtórzeniowe

Lekcja 4. Operatory logiczne i rzutowanie.

Witaj w materiałach powtórzeniowych przygotowanych specjalnie dla Ciebie. Znajdziesz tutaj powtórkę materiału z ostatniej lekcji oraz zadania do wykonania dla chętnych. Z pewnością sobie poradzisz. Powodzenia!

Cel lekcji

Celem lekcji jest przedstawienie rzutowania, operatorów matematycznych i logicznych oraz zaprezentowanie mechanizmów inkrementacji i dekrementacji w języku C#.

Rzutowanie jest procesem tymczasowej zmiany typu liczbowego na inny zbliżony typ.

Dotychczas przy potrzebie zmiany typu wprowadzonych przez użytkownika danych, lub wypisywanych na ekran stosowaliśmy **parsowanie**. Aby przekonwertować tekst na dany typ (np. string na int), należało skorzystać z polecenie int.Parse(tekst); Aby zamienić dany typ na tekst, należy skorzystać z polecenie **.ToString().** Co natomiast, kiedy chcemy zmienić typy liczbowe, przykładowo **float na int? Albo float na double?** W takim przypadku możemy skorzystać z mechanizmu, który nazywa się **rzutowaniem**.

W przypadku zmiennych nierzadko zdarza się, że zmienne posiadają nieprawidłowy typ, z punktu widzenia działania, które chcemy wykonać. Weźmy za przykład operację obliczania średniej. Gdy uczeń ma 5 ocen, których suma wynosi 23, średnia wyjdzie 4.6. Jeżeli wykonamy operację obliczania średniej na liczbach całkowitych, to niestety ale wynik zostanie zafałszowany, przez usunięcie części po przecinku. Nie pomoże też zmiana typu wyniku na zmiennoprzecinkowy, gdyż same obliczenia wykonają się nadal dla liczbach całkowitych. Aby również operacje wykonały się w pożądany sposób, możemy tymczasowo wymusić zmianę ich typu, stosując rzutowanie. Rzutowanie polega na dopisaniu pożądanego typu w nawiasie, przed zmienną, której typ ma zostać zmieniony. Komputer wtedy będzie traktował ją na czas obliczeń, jako wartość o innym typie.

Typ int (całkowity)	Typ float (zmiennoprzec.)	Typ float rzutowany

`int liczbaOcen = 5;`
`int sumaOcen = 23;`
`int srednia = sumaOcen/liczbaOcen;`
	`int liczbaOcen = 5;`
	`int sumaOcen = 23;`
	`float srednia = sumaOcen / liczbaOcen`
		`int liczbaOcen = 5;`
		`int sumaOcen = 23;`
		`float srednia = (float)sumaOcen / (float)liczbaOcen;`

wynik = 4	**wynik = 4**	**wynik = 4.6**

Operatory porównania i relacji to nic innego jak elementy znane doskonale z matematyki, pozwalające na porównywanie dwóch wartości. Do podstawowych operatorów zaliczamy:

- znak większości (>),
- mniejszości (<),
- większe lub równe (>=),
- mniejsze lub równe (<=),
- równe (==) oraz różne (!=).

Wyróżniamy również operatory logiczne:

- Logiczne „i” (**&&**) (wszystkie warunki muszą być spełnione)
- Logiczne „lub” (**||**) (jeden z warunków musi być spełniony)
- Negacja (wykrzyknik) **!**true = false; (zmienia wartość logiczną na przeciwną)

Przy ich pomocy możemy zbudować proste i złożone zdania logiczne, których wartość wynosi **prawda** lub **fałsz** (true/false). Wynik sprawdzania możemy przypisać do jednego z poznanych typów zmiennych - **typ bool**.

Przykład użycia:

```
bool wynik = 5 > 3;
Console.WriteLine(wynik);
wynik = "ala" == "ma kota";
Console.WriteLine(wynik);
```

Ważne:

- Operatora równości == i operator przypisania =, to dwie różne rzeczy. Dwie instrukcje x = y oraz x == y to dwie różne operacje.
- Porównywać możemy tylko wartości tego samego typu. Nie można porównać np. zmiennej typu string = „Ala” ze zmienną typu int = 5. Jest to nielogiczne.

Przykłady zdań logicznych:

```
int a = 5;
int b =7;
bool czyRowne = a == b; //false
bool czyRozne = a != b; //true
bool czyWieksza = a > b //false;
bool czyMniejsza = a < b; // true;
```

Operator trójargumentowy (wyrażenie warunkowe) pozwala pokazać dowolny tekst jako wynik, zamiast wyświetlania wartości True/False**.** Jego składnia to:

[wyrażenie_logiczne] ? [wykonaj to, gdy prawda] : [wykonaj to, gdy fałsz]

Przykład użycia:

```
string wynik = czyProstokatny ? "Tak" : "Nie";
Console.WriteLine("Czy trójkąt jest prostokątny? -> " + wynik);
```

Inkrementacja i dekrementacja.

W programowaniu wykonujemy wiele operacji matematycznych. Jako, że programiści mają tendencję do upraszczania sobie życia możemy spotkać się, ze skracaniem niektórych operacji do kilku znaków.

Inkrementacja (czyli zwiększanie) lub **dekrementacja** (zmniejszanie) wartości o jeden jest jedną z takich operacji. zapis. Przy pomocy operatora **+=** dodajemy nową wartość do naszej zmiennej pomijając odwołanie się do niej po prawej stronie zapisu. Jeszcze większym skróceniem jest zastosowanie symbolu **++**.

Standardowe zwiększanie wartości	**Inkrementacja**	**Skrócona inkrementacja**
liczba = liczba + 1;	liczba += 1;	liczba++;

Analogicznie sytuacja wygląda w przypadku dekrementacji.

Standardowe zwiększanie wartości	**Inkrementacja**	**Skrócona inkrementacja**
liczba = liczba - 1;	liczba -= 1;	liczba--;

Operatorów inkrementacji i dekrementacji używa się **tylko na typach liczbowych**, czyli int, float, double itd. Będziemy ich dużo stosować w pętlach, które poznamy na jednych z kolejnych zajęć. **Operator inkrementacji i dekrementacji może występować przed lub po zmiennej.** Gdy występuje przed zmienną, zmiana jej wartości nastąpi w danej instrukcji. Jeżeli występuje po zmiennej, zmiana wartość wystąpi dopiero w kolejnej instrukcji.

Przykładowe użycie:

```
int liczba = 8;
Console.WriteLine("Wartość liczby {0}", liczba);
liczba++;
liczba++;
Console.WriteLine("Wartość po inkrementacji {0}", liczba);
liczba--;
liczba--;
liczba--;
Console.WriteLine("Wartość po dekrementacji {0}", liczba);
Console.ReadKey();
```

Dodatkowa misja Programisty:

Zadanie 1

Załóżmy, że a=5, b=9. Utwórz 2 zmienne typu bool o nazwie operacja1, operacja2 i przypisz do nich dowolną operację matematyczną na zmiennych a i b w taki sposób, aby zmienna operacja1 zwracała wartość true, a zmienna operacja2 zwracała wartość false. Wypisz wartość wszystkich zmiennych w konsoli.

Przykładowe częściowe rozwiązanie:

bool operacja1 = (a>b) //zwróci wartość false, bo a jest mniejsze od b

Zadanie 2

Napisz program, który obliczy cenę zużytego materiału (filamentu) podczas druku konkretnego modelu na drukarce 3D. Koszt obliczysz ze wzoru:

(koszt Szpuli Filamentu * waga Figurki)/1000

Wersja rozszerzona

Dodaj do programu obliczanie ceny zużytego prądu przez drukarkę podczas wydruku modelu. Zadeklaruj stałą, do której przypiszesz średnią stawkę za 1kW. Wspomóż się wzorami:

zużycie prądu = **(moc drukarki * czas wydruku) /1000**

koszt prądu = **zużycie prądu * stawka za 1 kW**